

Knowledge Representation for the Semantic Web

Winter Quarter 2011

Slides 8 – 02/22/2011

Pascal Hitzler

Kno.e.sis Center

Wright State University, Dayton, OH

<http://www.knoesis.org/pascal/>



Textbook (required)

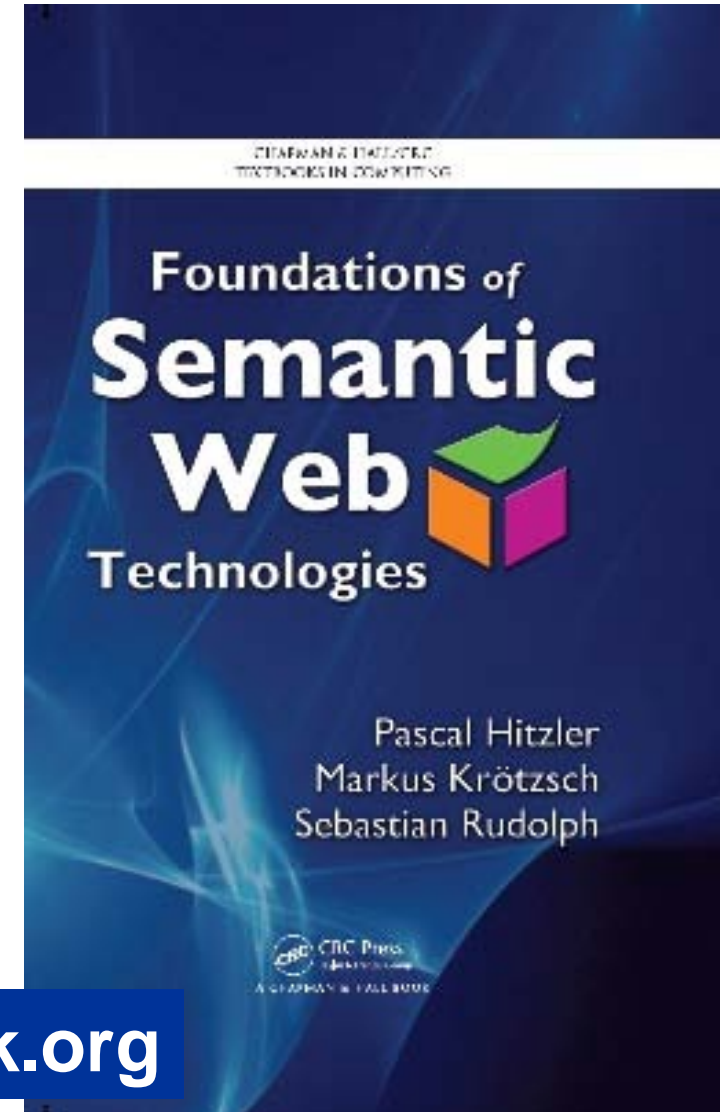
**Pascal Hitzler, Markus Krötzsch,
Sebastian Rudolph**

**Foundations of Semantic Web
Technologies**

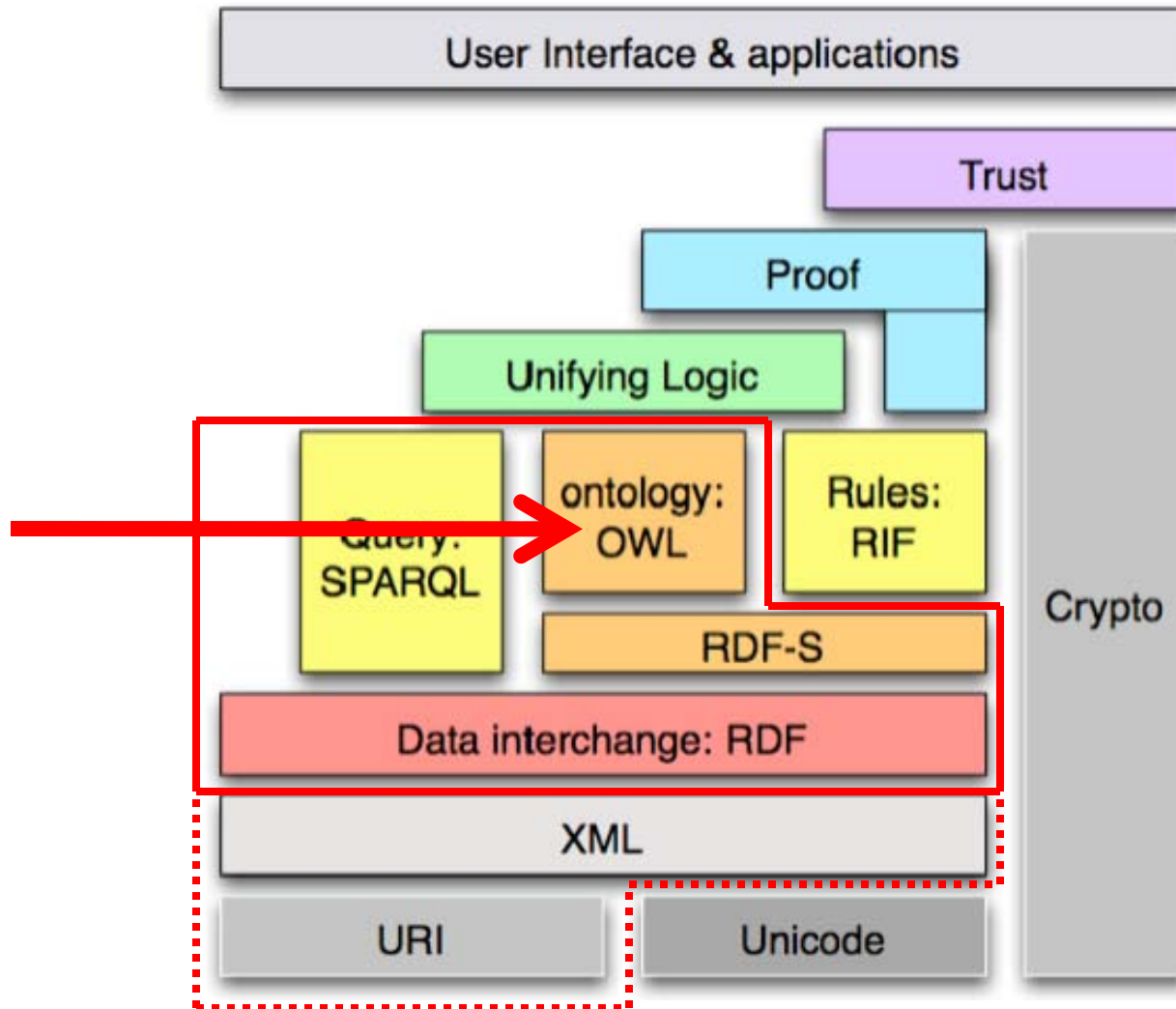
Chapman & Hall/CRC, 2010

**Choice Magazine Outstanding Academic
Title 2010 (one out of seven in Information
& Computer Science)**

<http://www.semantic-web-book.org>



Today: OWL Syntax



```
ex:speaksWith    rdfs:domain    ex:Homo .  
ex:Homo         rdfs:subClassOf  ex:Primates .
```

does not RDFS-entail

```
ex:speaksWith    rdfs:domain    ex:Primates .
```

although it is a valid OWL entailment.

It does RDFS-entail

```
rdfs:subClassOf  rdf:type  rdf:Property
```

which is not a valid OWL entailment.

- **RDF/XML Syntax**
 - The only *normative* syntax (i.e. to be OWL 2 compliant, a tool has to support this (and only this) syntax).
- **Turtle Syntax**
 - Straightforward Turtle version of the RDF/XML Syntax.
 - We will cover the RDF Syntax using Turtle or RDF/XML.
- **Functional Style Syntax**
 - Prefix-syntax, given as formal grammar
 - Clean, adjustable, modifiable, easily parsable
 - Used for *defining* OWL 2 in the W3C Specs.
- **Manchester Syntax**
 - User-friendly(?) syntax, used e.g. in Protégé 4
- **OWL/XML Syntax**
 - Notational variant of the Functional Style Syntax.
 - Does not use RDF triples, but simply XML tree structure.

- **Many examples, translated into all syntaxes:**
- **Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, Sebastian Rudolph**
OWL 2 Web Ontology Language: Primer.
W3C Recommendation, 27 October 2009.
<http://www.w3.org/TR/owl2-primer/>

1. **RDF Syntax**
2. **Other Syntaxes and OWL Variants**
3. **Class Project**
4. **Class Presentations**

- `:mary rdfs:type :Person .`
 - `:Mother rdfs:subClassOf :Woman .`
 - `:john :hasWife :Mary .`
 - `:hasWife rdfs:subPropertyOf :hasSpouse`
 - `:hasWife rdfs:range :Woman .`
 - `:hasWife rdfs:domain :Man .`
 - `owl:Thing`
 - `owl:Nothing`
 - `owl:topProperty`
 - `owl:bottomProperty`
- `Person(mary)`
 - `Mother \sqsubseteq Woman`
 - `hasWife(john,mary)`
 - `hasWife \sqsubseteq hasSpouse`
 - `$\top \sqsubseteq \forall \text{hasWife}. \text{Woman}$`
 - `$\top \sqsubseteq \forall \text{hasWife}^{-}. \text{Man}$ or $\exists \text{hasWife}. \top \sqsubseteq \text{Man}$`
 - `\top`
 - `\perp`
 - `U`
 - `?`

owl namespace: <http://www.w3.org/2002/07/owl#>

- **ABox assignments of individuals to classes or properties**
- **ALC:** \sqsubseteq, \equiv for classes
 $\sqcap, \sqcup, \neg, \exists, \forall$
 \top, \perp
- **SR:** + property chains, property characteristics, role hierarchies \sqsubseteq
- **SRO:** + nominals $\{o\}$
- **SROI:** + inverse properties
- **SROIQ:** + qualified cardinality constraints
- **SROIQ(D):** + datatypes (including facets)

- + top and bottom roles (for objects and datatypes)
- + disjoint properties
- + Self
- + Keys (not in SROIQ(D), but in OWL)

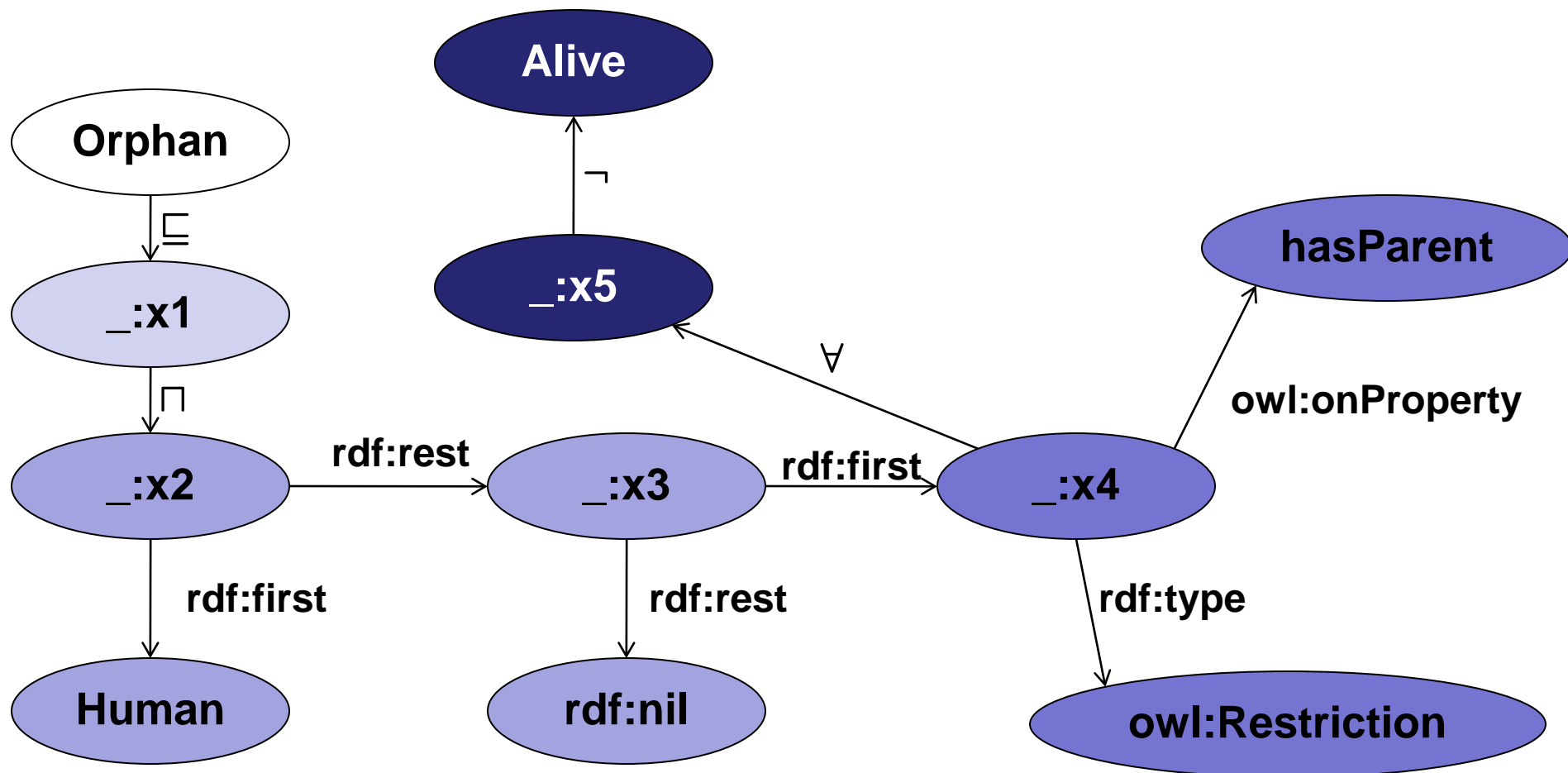
- How do you put SROIQ(D) axioms like

```
Orphan  $\sqsubseteq$  Human  $\sqcap$   $\forall$ hasParent. $\neg$ Alive
```

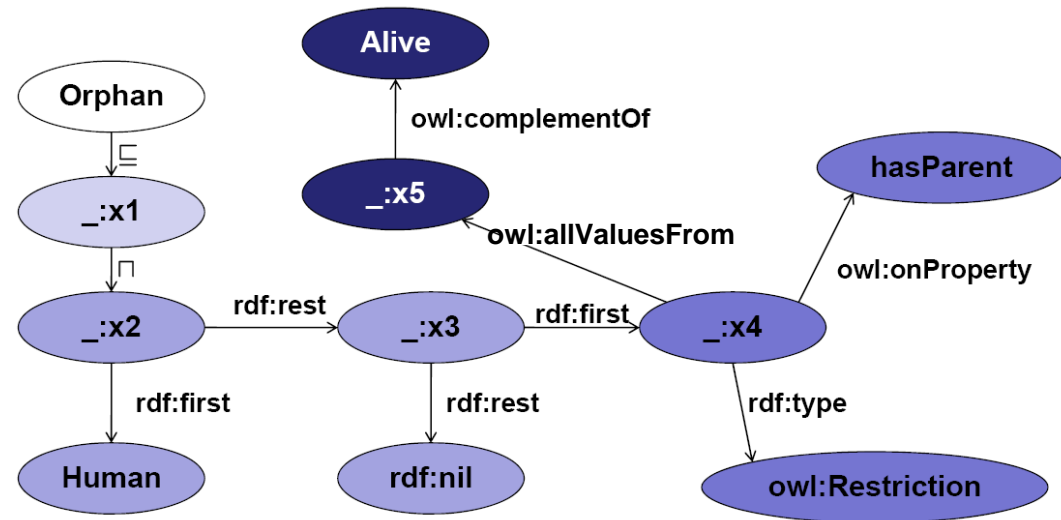
into a graph structure?

- How do you do it such that the RDF Schema semantics and the DL semantics are not violated?
- How do you do it without violating the main conceptual ideas behind RDF and DLs?
- **That's actually impossible without violating either RDF or DL. We have to do some approximations, and accept that the layering cannot be perfect.**

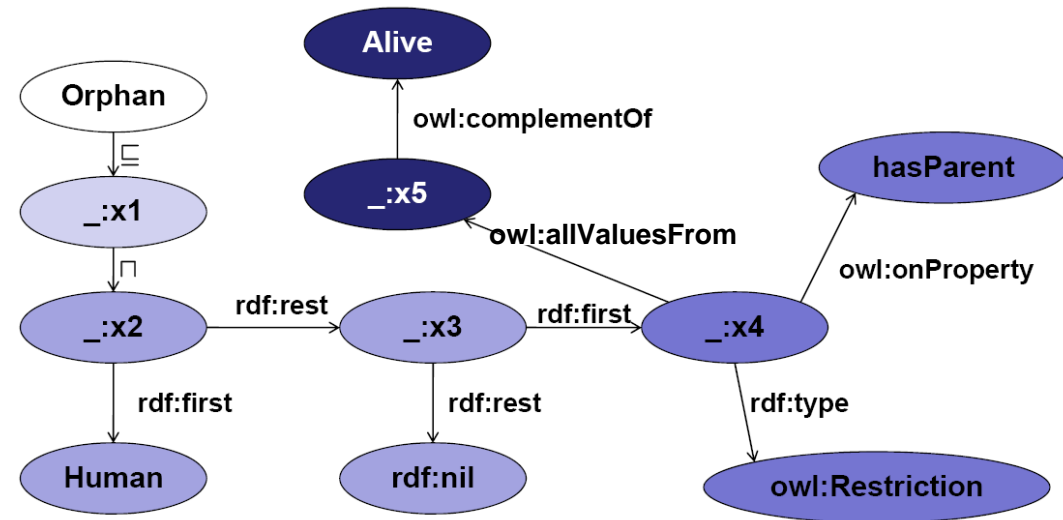
$\text{Orphan} \sqsubseteq \text{Human} \sqcap \forall \text{hasParent}.\neg \text{Alive}$



- From here on, you can basically make the RDF syntax yourself.
- You only need to know the OWL vocabulary to use and some constructs need some design decisions, which are sometimes almost arbitrary.



- You get all kinds of entailments which are entirely irrelevant for the OWL knowledge base.



- `owl:complementOf rdf:type rdf:Property .`
- `_:x5 owl:complementOf _:xyz .`
- `owl:Restriction rdf:type rdfs:Class .`
- `:hasParent rdf:type rdfs:Resource .`
- `owl:Restriction rdfs:subClassOf rdfs:Resource .`
- `owl:Restriction rdfs:subClassOf owl:Restriction .`

OWL RDF Syntax: Individuals

```
:Mary rdf:type :Woman .
```

```
:John :hasWife :Mary .
```

```
:John owl:differentFrom :Bill .
```

$\{John\} \sqcap \{Bill\} \sqsubseteq \perp$

```
:James owl:sameAs :Jim.
```

$\{John\} \equiv \{Jim\}$

```
:John :hasAge "51"^^xsd:nonNegativeInteger .
```

```
[ ] rdf:type owl:NegativePropertyAssertion ;  
    owl:sourceIndividual :Bill ;  
    owl:assertionProperty :hasWife ;  
    owl:targetIndividual :Mary .
```

$\neg\text{hasWife}(Bill, Mary)$

```
[ ] rdf:type owl:NegativePropertyAssertion ;  
    owl:sourceIndividual :Jack ;  
    owl:assertionProperty :hasAge ;  
    owl:targetValue 53 .
```

```
:Woman rdfs:subClassOf :Person .
```

```
:Person owl:equivalentClass :Human .
```

```
[ ] rdf:type owl:AllDisjointClasses ;  
    owl:members ( :Woman :Man ) .
```

Woman \sqcap Man $\sqsubseteq \perp$

```
:hasWife rdfs:subPropertyOf :hasSpouse .
```

```
:hasWife rdfs:domain :Man ;  
         rdfs:range :Woman .
```

OWL RDF Syntax: Complex Classes

```
:Mother owl:equivalentClass [  
  rdf:type owl:Class ;  
  owl:intersectionOf ( :Woman :Parent )  
] .
```

Mother \equiv Woman \sqcap Parent

```
:Parent owl:equivalentClass [  
  rdf:type owl:Class ;  
  owl:unionOf ( :Mother :Father )  
] .
```

Parent \equiv Mother \sqcup Father

```
:ChildlessPerson owl:equivalentClass [  
  rdf:type owl:Class ;  
  owl:intersectionOf ( :Person [ owl:complementOf :Parent ] )  
] .
```

ChildlessPerson \equiv Person \sqcap \neg Parent

```
:Grandfather rdfs:subClassOf [  
  rdf:type owl:Class ;  
  owl:intersectionOf ( :Man :Parent )  
] .
```



```
:Jack rdf:type [
  rdf:type          owl:Class ;
  owl:intersectionOf ( :Person
                        [ rdf:type          owl:Class ;
                          owl:complementOf :Parent      ]
                        )
] .
```

Person \sqcap \neg Parent (Jack)

```
:Parent owl:equivalentClass [  
  rdf:type owl:Restriction ;  
  owl:onProperty :hasChild ;  
  owl:someValuesFrom :Person  
] .
```

Parent $\equiv \exists \text{hasChild}.\text{Person}$

```
:Orphan owl:equivalentClass [  
  rdf:type owl:Restriction ;  
  owl:onProperty [ owl:inverseOf :hasChild ] ;  
  owl:allValuesFrom :Dead  
] .
```

Orphan $\equiv \forall \text{hasChild}^{\neg}.\text{Dead}$

```
:JohnsChildren owl:equivalentClass [  
  rdf:type owl:Restriction ;  
  owl:onProperty :hasParent ;  
  owl:hasValue :John  
]
```

JohnsChildren $\equiv \exists$ hasParent.{John}

```
:NarcisticPerson owl:equivalentClass [  
  rdf:type owl:Restriction ;  
  owl:onProperty :loves ;  
  owl:hasSelf "true"^^xsd:boolean .  
]
```

NarcisticPerson $\equiv \exists$ loves.Self

OWL RDF Syntax: Restrictions

≤ 4 hasChild.Parent (John)

```
:John rdf:type [
  rdf:type                owl:Restriction ;
  owl:maxQualifiedCardinality "4"^^xsd:nonNegativeInteger ;
  owl:onProperty         :hasChild ;
  owl:onClass            :Parent
] .
```

```
:John rdf:type [
  rdf:type                owl:Restriction ;
  owl:minQualifiedCardinality "2"^^xsd:nonNegativeInteger ;
  owl:onProperty         :hasChild ;
  owl:onClass            :Parent
] .
```

≥ 2 hasChild.Parent (John)

```
:John rdf:type [
  rdf:type                owl:Restriction ;
  owl:qualifiedCardinality "3"^^xsd:nonNegativeInteger ;
  owl:onProperty         :hasChild ;
  owl:onClass            :Parent
] .
```

=3 hasChild.Parent (John)

```
:John rdf:type [
  rdf:type          owl:Restriction ;
  owl:cardinality "5"^^xsd:nonNegativeInteger ;
  owl:onProperty  :hasChild
] .
```

=5 hasChild.⊤ (John)

```
:MyBirthdayGuests owl:equivalentClass [
  rdf:type          owl:Class ;
  owl:oneOf       ( :Bill   :John   :Mary )
] .
```

MyBirthdayGuests ≡ {Bill, John, Mary}

OWL RDF Syntax: Properties

```
:hasParent owl:inverseOf :hasChild .
```

```
:Orphan owl:equivalentClass [  
  rdf:type owl:Restriction ;  
  owl:onProperty [ owl:complementOf :hasChild ] ;  
  owl:allValuesFrom :Dead  
] .
```

Orphan $\equiv \forall \text{hasChild}^c . \text{Dead}$

```
:hasSpouse rdf:type owl:SymmetricProperty .
```

```
:hasChild rdf:type owl:AsymmetricProperty .
```

```
:hasParent owl:propertyDisjointWith :hasSpouse .
```

```
:hasRelative rdf:type owl:ReflexiveProperty .
```

```
:parentOf rdf:type owl:IrreflexiveProperty .
```

```
:hasHusband rdf:type owl:FunctionalProperty .
```

```
:hasHusband rdf:type owl:InverseFunctionalProperty .
```

```
:hasAncestor rdf:type owl:TransitiveProperty .
```

```
:hasGrandparent owl:propertyChainAxiom ( :hasParent :hasParent ).
```

hasParent o hasParent \sqsubseteq hasGrandParent

```
:Person owl:hasKey ( :hasSSN ) .
```

In OWL 2 a collection of (data or object) properties can be assigned as a key to a class expression. This means that each named instance of the class expression is uniquely identified by the set of values which these properties attain in relation to the instance.

```
:personAge owl:equivalentClass
```

```
  [ rdf:type rdfs:Datatype;
```

```
    owl:onDatatype xsd:integer;
```

Datatype *facets*

```
    owl:withRestrictions (
```

```
      [ xsd:minInclusive "0"^^xsd:integer ]
```

```
      [ xsd:maxInclusive "150"^^xsd:integer ]
```

```
    )
```

```
  ] .
```

```
:majorAge owl:equivalentClass
```

```
  [ rdf:type rdfs:Datatype;
```

```
    owl:intersectionOf (
```

```
      :personAge
```

```
      [ rdf:type rdfs:Datatype;
```

```
        owl:datatypeComplementOf :minorAge ]
```

```
    )
```

```
  ] .
```

```
:toddlerAge owl:equivalentClass
```

```
  [ rdf:type rdfs:Datatype;
```

```
    owl:oneOf ( "1"^^xsd:integer "2"^^xsd:integer )
```

```
  ] .
```


Essential OWL Features

Feature	Related OWL vocabulary	FOL	DL
top/bottom class	<code>owl:Thing/owl:Nothing</code>	(axiomatise)	\top/\perp
Class intersection	<code>owl:intersectionOf</code>	\wedge	\sqcap
Class union	<code>owl:unionOf</code>	\vee	\sqcup
Class complement	<code>owl:complementOf</code>	\neg	\neg
Enumerated class	<code>owl:oneOf</code>	(ax. with \approx)	$\{a\}$
Property restrictions	<code>owl:onProperty</code>		
Existential	<code>owl:someValueFrom</code>	$\exists y \dots$	$\exists R.C$
Universal	<code>owl:allValuesFrom</code>	$\forall y \dots$	$\forall R.C$
Min. cardinality	<code>owl:minQualifiedCardinality</code> <code>owl:onClass</code>	$\exists y_1 \dots y_n. \dots$	$\geq n$ S.C
Max. cardinality	<code>owl:maxQualifiedCardinality</code> <code>owl:onClass</code>	$\forall y_1 \dots y_{n+1}. \dots \rightarrow \dots$	$\leq n$ S.C
Local reflexivity	<code>owl:hasSelf</code>	$R(x,x)$	$\exists R.Self$

Essential OWL Features

Feature	Related OWL vocabulary	DL	
Property chain	<code>owl:propertyChainAxiom</code>	◦	
Inverse	<code>owl:inverseOf</code>	R^-	
Key	<code>owl:hasKey</code>		
Property disjointness	<code>owl:propertyDisjointWith</code>	$\text{Dis}(R,S)$	
Property characteristics	<code>rdf:type</code>		
Symmetric	<code>owl:SymmetricProperty</code>	$\text{Sym}(R)$	
Asymmetric	<code>owl:AsymmetricProperty</code>	$\text{Asy}(R)$	
Reflexive	<code>owl:ReflexiveProperty</code>	$\text{Ref}(R)$	
Irreflexive	<code>owl:IrreflexiveProperty</code>	$\text{Irr}(R)$	
Transitivity	<code>owl:TransitiveProperty</code>	$\text{Tra}(R)$	
Subclass	<code>rdfs:subClassOf</code>	$\forall x.C(x) \rightarrow D(x)$	$C \sqsubseteq D$
Subproperty	<code>rdfs:subPropertyOf</code>	$\forall x,y.R(x,y) \rightarrow S(x,y)$	$R \sqsubseteq S$

```
<http://example.com/owl/families> rdf:type owl:Ontology .
```

```
@prefix : <http://example.com/owl/families/> .  
@prefix otherOnt: <http://example.org/otherOntologies/families/> .  
@prefix owl: <http://www.w3.org/2002/07/owl#> .  
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .  
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .  
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
<http://example.com/owl/families>  
owl:imports <http://example.org/otherOntologies/families/> .
```

```
@prefix : <http://example.com/owl/families/> .
@prefix otherOnt: <http://example.org/otherOntologies/families/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
<http://example.com/owl/families>
owl:imports <http://example.org/otherOntologies/families/> .
```

```
:Mary owl:sameAs otherOnt:MaryBrown .
:John owl:sameAs otherOnt:JohnBrown .
:Adult owl:equivalentClass otherOnt:Grownup .
:hasChild owl:equivalentProperty otherOnt:child .
:hasAge owl:equivalentProperty otherOnt:age .
```

Each class, property, or individual needs to be declared.

```
:John      rdf:type  owl:NamedIndividual .
:Person    rdf:type  owl:Class .
:hasWife   rdf:type  owl:ObjectProperty .
:hasAge    rdf:type  owl:DatatypeProperty .
```

Punning:

Same URI can stand e.g. for both an individual and a class:

```
:John      rdf:type      :Father .
:Father    rdf:type      :SocialRole .
```

Semantics: This is semantically interpreted as if the two occurrences of `Father` were in fact distinct.

Not allowed: E.g. use of a URI for both object and datatype property.

```
:Person rdfs:comment "Represents the set of all people."^^xsd:string .
```

```
:Man rdfs:subClassOf :Person .  
[] rdf:type owl:Axiom ;  
 owl:annotatedSource :Man ;  
 owl:annotatedProperty rdfs:subClassOf ;  
 owl:annotatedTarget :Person ;  
 rdfs:comment "States that every man is a person."^^xsd:string .
```

1. RDF Syntax
2. **Other Syntaxes and OWL Variants**
3. Class Project
4. Class Presentations

- **OWL 2 DL is the “description logic” version of OWL**
 - global restrictions from SROIQ(D) apply
 - RDF can only be used in a very controlled fashion (only what is necessary for expressing OWL axioms)
 - model-theoretic semantics of SROIQ(D) is used, called *OWL 2 Direct Semantics*
- **OWL 2 Full is unrestricted OWL 2 DL plus all of RDF(S).**
 - no global restrictions
 - RDF can be used freely
 - semantics is a hybrid of RDFS and OWL 2 DL semantics, called *RDF-Based Semantics*
- **Both semantics are in the W3C recommendation.**
No implementations of the OWL 2 Full semantics exist.

- **The OWL 2 spec describes three profiles (fragments, sublanguages) which have polynomial complexity.**
 - **OWL EL (the description logic EL++)
forthcoming class presentation**
 - **OWL QL (the description logic DL Lite_R)
forthcoming class presentation**
 - **OWL RL (the description logic DLP)
skipped**
 - **inspired by intersecting OWL with Datalog**
 - **implemented e.g. in Oracle 11g**

```
SubClassOf (  
  :ChildlessPerson  
ObjectIntersectionOf (  
  :Person  
ObjectComplementOf (  
  ObjectSomeValuesFrom (  
    ObjectInverseOf ( :hasParent )  
    owl:Thing  
  )  
)  
)  
)
```

$\text{ChildlessPerson} \sqsubseteq \text{Person} \sqcap \neg \exists \text{hasParent} \bar{\top}$

```
ClassAssertion (  
  ObjectIntersectionOf (  
    :Person  
    ObjectComplementOf ( :Parent )  
  )  
  :Jack  
)
```

$\text{Person} \sqcap \neg \text{Parent} (\text{Jack})$

```
Class: Parent
  EquivalentTo: hasChild some Person
  EquivalentTo: Mother or Father
```

```
Class: HappyPerson
  EquivalentTo: hasChild only Happy and hasChild some Happy
Class: JohnsChildren
  EquivalentTo: hasParent value John
Class: NarcisticPerson
  EquivalentTo: loves Self
Class: Orphan
  EquivalentTo: inverse hasChild only Dead
Class: Teenager
  SubClassOf: hasAge some integer[<= 13 , >= 19]
```

```
Class: X
  SubClassOf: Parent and hasChild max 1 and hasChild only Female
  EquivalentTo: {Mary, Bill, Meg} and Female
```

```
Individual: John
Types: Father
Types: hasChild max 4 Parent
Types: hasChild min 2 Parent
Types: hasChild exactly 3 Parent
Types: hasChild exactly 5
Facts: hasAge "51"^^xsd:integer
Facts: hasWife Mary
DifferentFrom: Bill
```

1. RDF Syntax
2. Other Syntaxes and OWL Variants
3. **Class Project**
4. Class Presentations

- **Make sure your ontology is actually in OWL DL (i.e. remove or remodel things which are not in OWL DL).**
- **Put your OWL ontology into Protégé:**

<http://protege.stanford.edu/>
- **Send the ontology serialized by Protégé (in RDF/XML or Turtle) to me by 8th of March.**

1. RDF Syntax
2. Other Syntaxes and OWL Variants
3. Class Project
4. **Class Presentations**

- **20 minutes (timing will be strict) plus a few questions**
- **Content selection is up to you. Presentation must contain the key results from the paper you are presenting. It can also contain material related to these key results which are not in the paper itself (but which you may find more interesting than the rest of the paper).**

Example criteria – may not all be weighted equally:

- **Quality of slides**
- **Quality and effectiveness of explanations**
- **Quality of presentation style (use of verbal and body language, use of media, flexibility in case of interaction with audience, time management)**
- **Correctness of content**
- **Grade of reaching the audience and getting the content across**
- **How “interesting” the material is presented**
- **Competence in answering questions**